Embedded Linux Conference Europe 2012 (Barcelona - November 5-7)

Embedded Linux RADAR device

Taking advantage on Linaro tools and HTML5 AJAX real-time visualization

Agustí FONTQUERNI GORCHS

af@iseebcn.com

ISEE INTEGRATION SOFTWARE ELECTRONIC ENGINEERING

www.isee.biz

OVERVIEW

- Introduction
- Highlights
  - A/D Converter access
  - Optimization programming techniques
  - Optimization from compilers
  - HTML5 + AJAX
  - Developing time / costs
- Demo
- Future
- Summary

INTRODUCTION

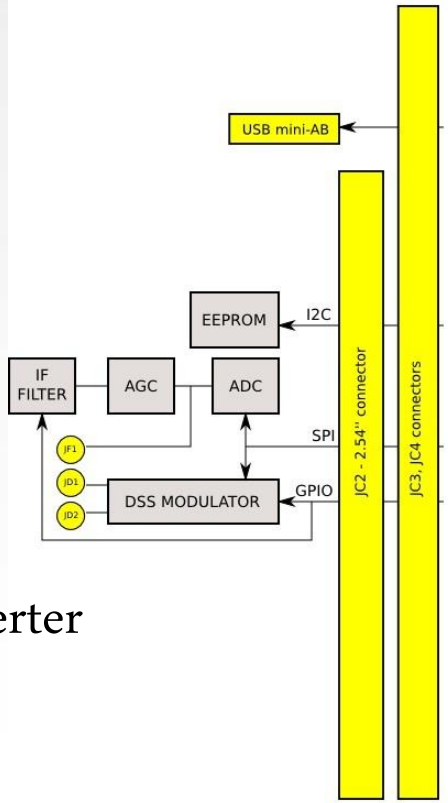
- hardware
  - **IGEPv2** board + **IGEP Radar** sensor

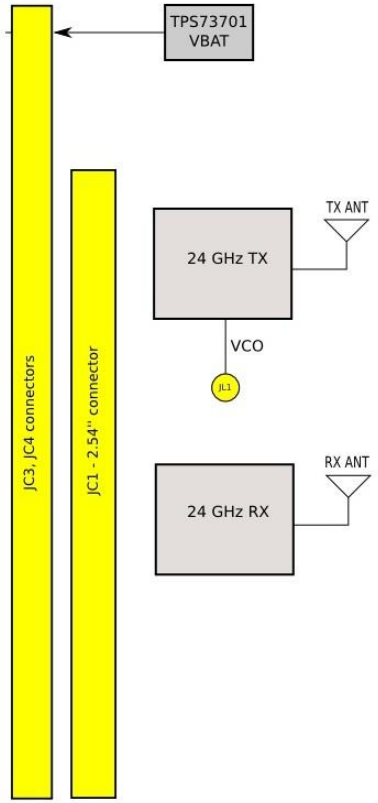
INTRODUCTION

- hardware
  - **IGEPv2** board
    - TI DM3730 processor
      - ARM Cortex A8 core at 1GHz
        - Processor : ARMv7 Processor rev 2 (v7l)
        - BogoMIPS : 996.74
      - DSP C64x+ core at 800Mhz
    - GPIO expansion J990 connector (BeagleBoard connector)
      - SPI bus
      - Input/output digital control lines
    - 5 Vcc Power

INTRODUCTION

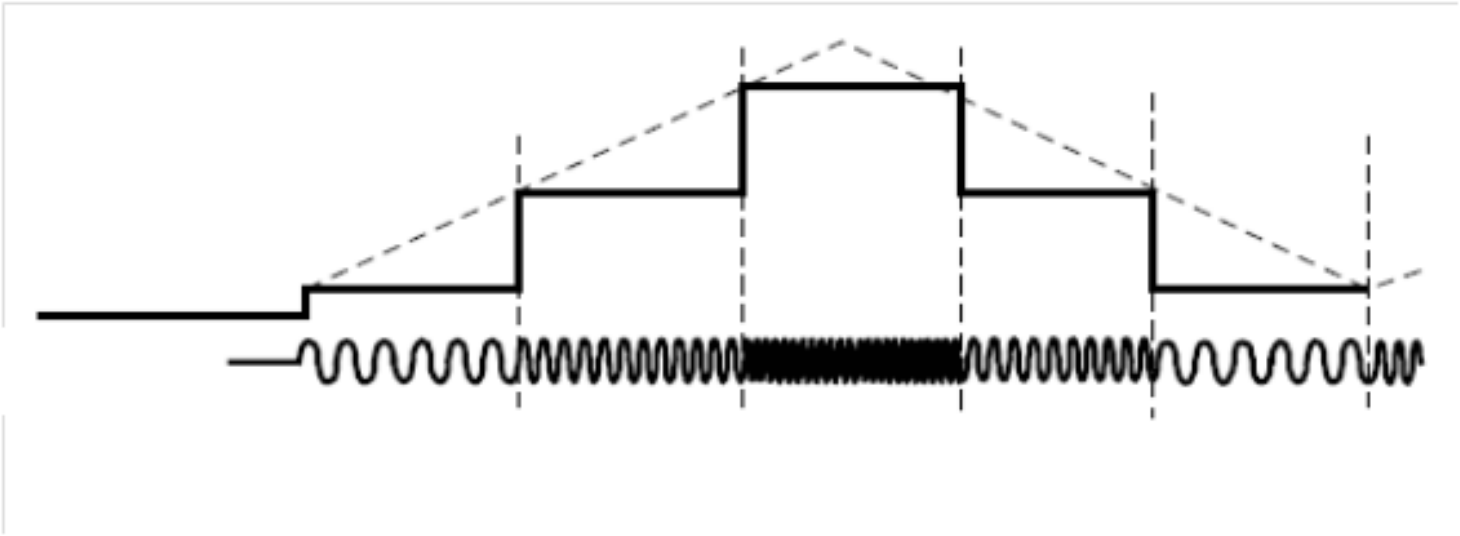
- hardware
  - **IGEP Radar** sensor
    - Antenna
    - RF Transmitter
    - RF Receiver
    - SPI DDS
    - SPI A/D Converter
    - PLL
    - IF Filter
    - I/O control lines





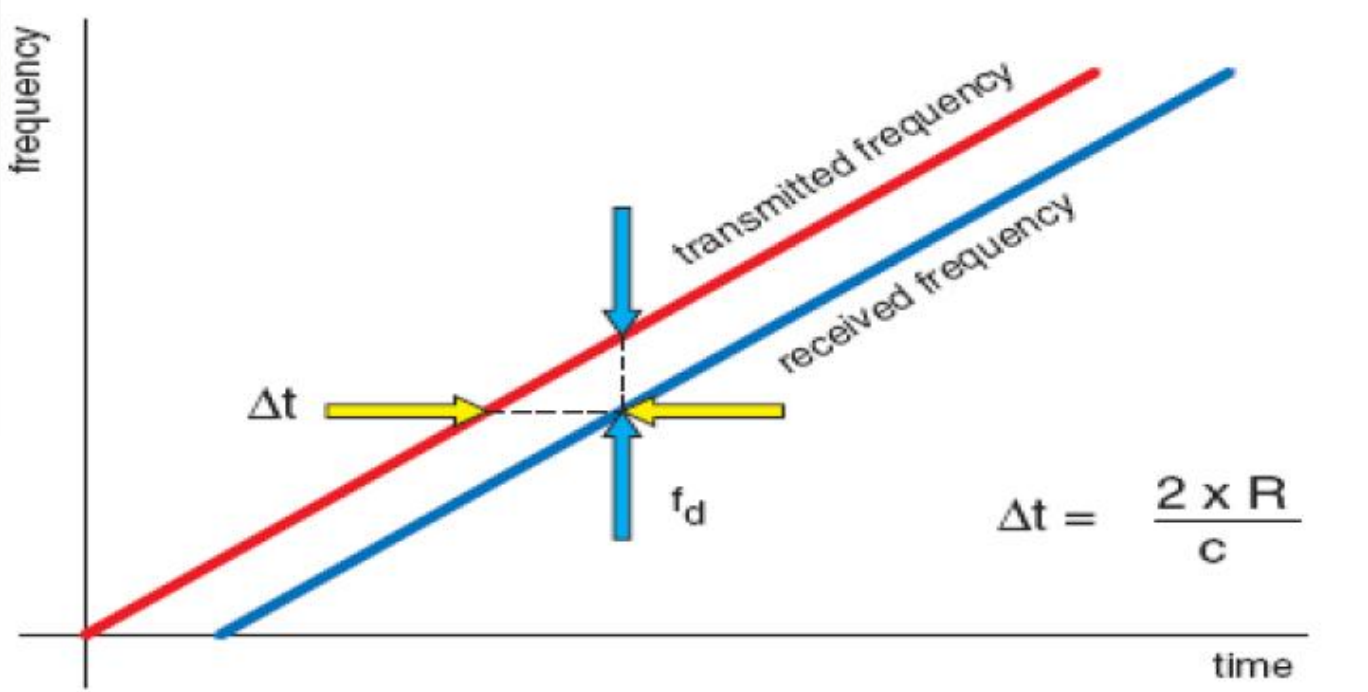
INTRODUCTION

- radar: (quick) FMCW Radar technology
  - FMCW: Frequency Modulation Constant Wave
    - Modulation from 24,0 Ghz to 24,25 Ghz (ISM K-Band)



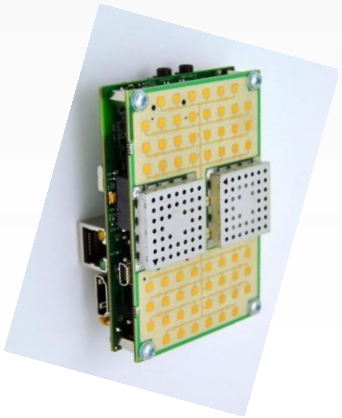
INTRODUCTION

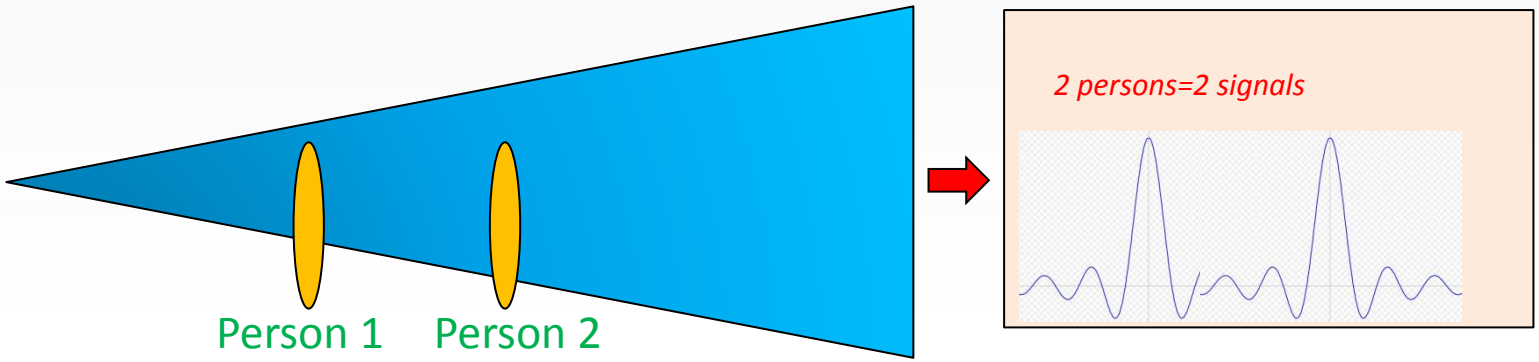
- radar: (quick) FMCW Radar technology
  - **diff frequency = measured distance**
  - balance between accuracy and latency/refresh measure



INTRODUCTION

- radar: (quick) FMCW Radar technology
  - it searches for positions of peaks of FFT in IF (Intermediate Frequency) signal that is proportional to distance measures





INTRODUCTION

- software
  - Linux **kernel 2.6.37**
    - Based on **TI BSP** with 2.6.37 kernel **fork**
    - The last for **OMAP3** (**dead line** NOT ported to mainline)
    - with **A LOT OF patches...**
      - from 2.6.37 official branch (kernel.org)
      - from ISEE to support IGEP devices (git.isee.biz)
  - Roofs based on **Poky 7.0.0** distribution and compiled with Yocto1.2 infrastructure
  - Specific **radar application** is developed on standard **C-code**

HIGHLIGHTS: A/D Converter access

- hwmon infrastructure
- comedy infrastructure
- spidev infrastructure (userspace / omapspi DMA transfer)
  - omapspi host driver **patch** (CS mode configuration)

```
diff --git a/drivers/spi/omap2_mcspi.c b/drivers/spi/omap2_mcspi.c
index 951a160..0a4a13d 100644
--- a/drivers/spi/omap2_mcspi.c
+++ b/drivers/spi/omap2_mcspi.c
@@ -252,7 +252,7 @@ static void omap2_mcspi_set_master_mode(struct
spi_master *master)
         l = mcspi_read_reg(master, OMAP2_MCSPI_MODULCTRL);
         MOD_REG_BIT(l, OMAP2_MCSPI_MODULCTRL_STEST, 0);
         MOD_REG_BIT(l, OMAP2_MCSPI_MODULCTRL_MS, 0);
-        MOD_REG_BIT(l, OMAP2_MCSPI_MODULCTRL_SINGLE, 1);
+        MOD_REG_BIT(l, OMAP2_MCSPI_MODULCTRL_SINGLE, 0);
         mcspi_write_reg(master, OMAP2_MCSPI_MODULCTRL, l);

         omap2_mcspi_ctx[master->bus_num - 1].modulctrl = l;
```

HIGHLIGHTS: A/D Converter access

- Linux **kernel module** for configuration of ADC121S101 spi-device A/D converter from user-space (opposite to igep00x0-board.c file)

```
#include <linux/module.h> /* Needed by all modules */
#include <linux/kernel.h> /* Needed for KERN_INFO */
#include <linux/device.h> /* Needed for 2.6.28 */
#include <linux/spi/spi.h>/* Needed for spi data structures */
#include <plat/mcspi.h>   /* Needed for OMAP3 spi driver */

static struct omap2_mcspi_device_config igep2_mcspi_config = {
        .turbo_mode  = 0,
        .single_channel     = 1,   // 0: slave, 1: master
};

static struct spi_board_info igep2_spi_board_info[]  = {
        { /* ADC IGEPv2: SPI4 CS0 */
                .modalias     = "spidev",
                .bus_num      = 4,
                .chip_select = 0,
                .max_speed_hz= 20000000,
                .controller_data     = &igep2_mcspi_config,
                .mode         = SPI_MODE_2,
        },
};
```

HIGHLIGHTS: A/D Converter access

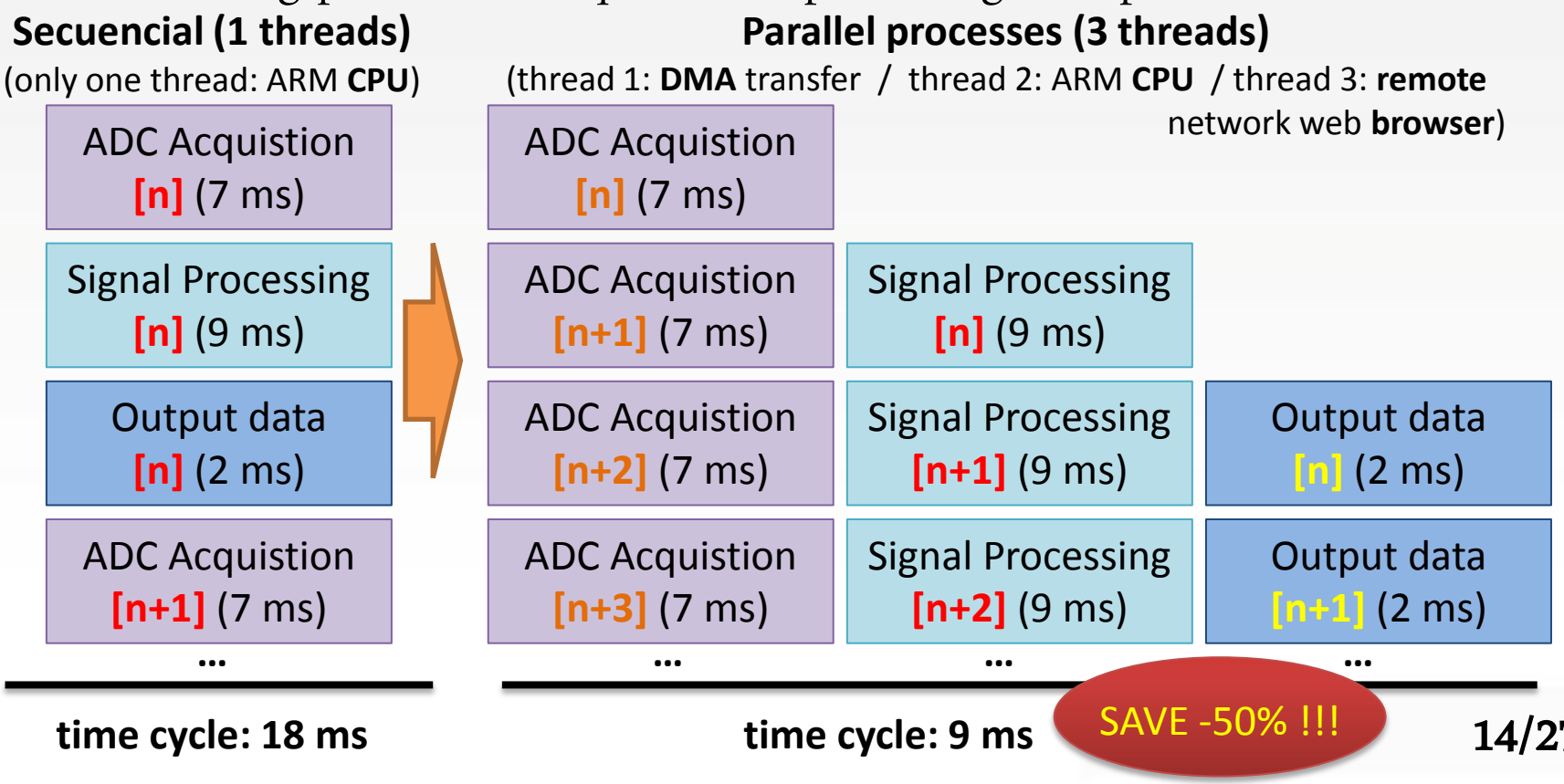
```
int radarspi_init(void)
{
	unsigned n;
	struct spi_master *master = NULL;
	struct spi_device *spidev = NULL;

	for (n = 0; n < ARRAY_SIZE(igep2_spi_board_info); n++ ) {
//extern struct spi_master *spi_busnum_to_master(u16 busnum);
		master = spi_busnum_to_master(igep2_spi_board_info[n].bus_num );

//extern struct spi_device * spi_new_device(struct spi_master *, struct spi_board_info *);
		spidev = spi_new_device(master, &igep2_spi_board_info[n]);
		spidevs[n] = spidev; //for remove spidev on radarspi_exit()
	}
	return 0;
}

void radarspi_exit(void)
{
	unsigned n;
	for (n = 0; n < ARRAY_SIZE(igep2_spi_board_info); n++ ) {
		spi_unregister_device(spidevs[n]);
	}
}

module_init(radarspi_init);
module_exit(radarspi_exit);

MODULE_AUTHOR("Agusti Fontquerni <afontquerni@iseebcn.com>");
MODULE_DESCRIPTION("IGEPv2 RADAR ADC SPI device driver ");
MODULE_LICENSE("GPL");
```

HIGHLIGHTS: Optimization programming techniques

- Debugging time bottlenecks
  - timeline marks
    - *gettimeofday*()
  - detection of the heaviest computing functions
    - Fast Fourier Transform (FFT) calculation
    - Data movements
    - Loops
  - detection of delay device response
    - acquisition time delay (A/D Converter wave data)
    - network TCP request and data response transfers

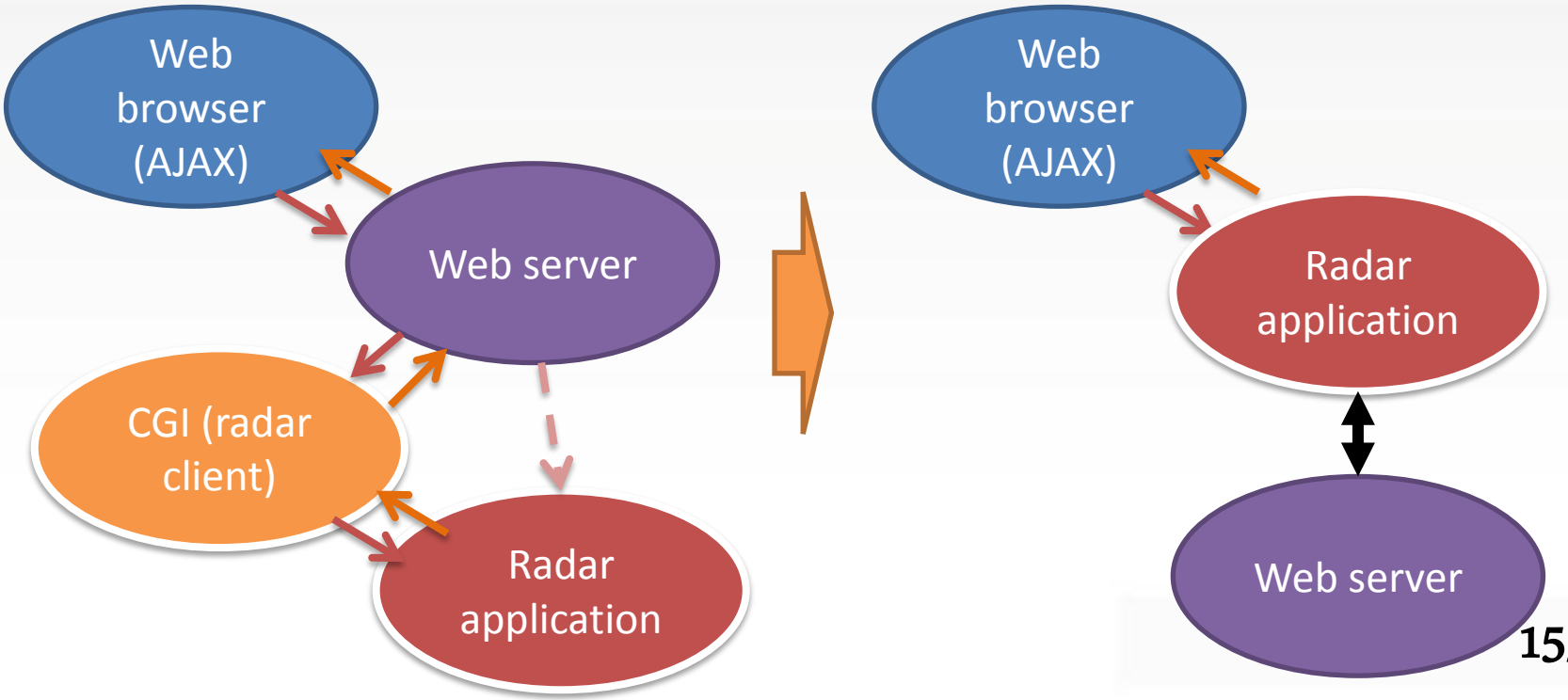
HIGHLIGHTS: Optimization programming techniques

- Paralleling processes (acquisition / processing / output data)



HIGHLIGHTS: Optimization programming techniques

- HTML5 data visualization: CGI vs 'Reverse Proxypass'
  - The **critical data path** (**JSON Data** for AJAX HTTP_Request)



HIGHLIGHTS: Optimization from compilers

- GCC 4.3.3 (poky 3.3.1)
- GCC 4.6.4 (yocto 1.2 / poky 7.0.0)
- GCC 4.5 (from www.linaro.org)
  - optimization GCC flags

http://www.linaro.org/linaro-blog/2011/10/25/compiler-flags-used-to-speed-up-linaro-android-2011-10-and-future-optimizations/

-march=**armv7-a**

-mtune=**cortex-a8**

-mfpu=**neon**

-mfloat-abi=**softfp**

HIGHLIGHTS: Optimization from compilers

- Comparative of compilers (binary size + radar process time)

COMPILER	BINARY SIZE	EXECUTION TIME
GCC 4.3.3 (poky 3.3.1)	212.0 KB	19 ms
GCC 4.6.4 (poky 7.0.0)	216.5 KB	9 ms
GCC 4.6 (www.linaro.org)	209.0 KB	9 ms

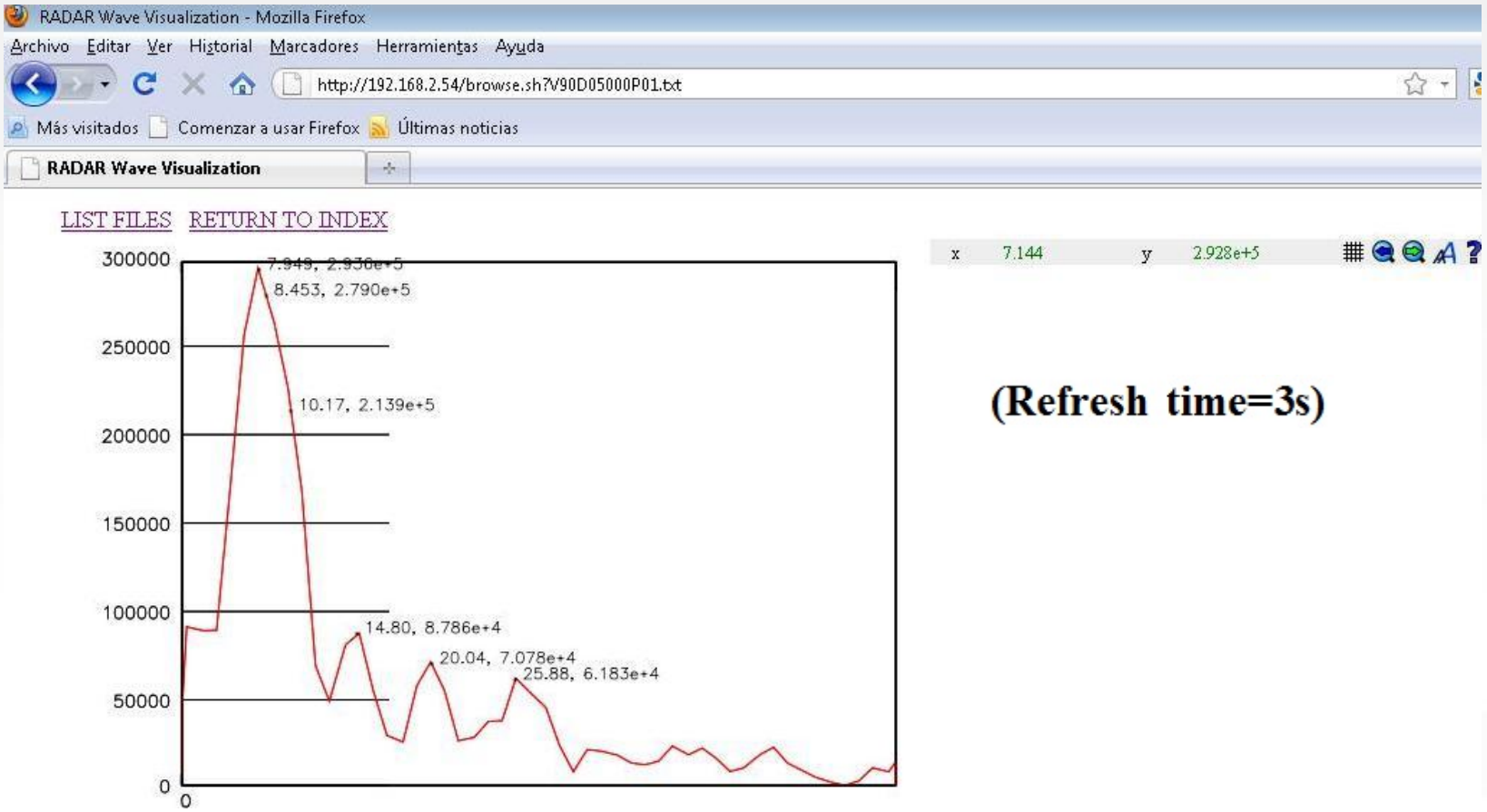
SAVE -53% !!!

HIGHLIGHTS: HTML5 + AJAX Visualization

- Gnuplot 'png image output' (NO HTML5)
- **Gnuplot** 'canvas output'
- **flop** (jQuery extension) – a Javascript plot library
- **Canvas** HTML5 + AJAX

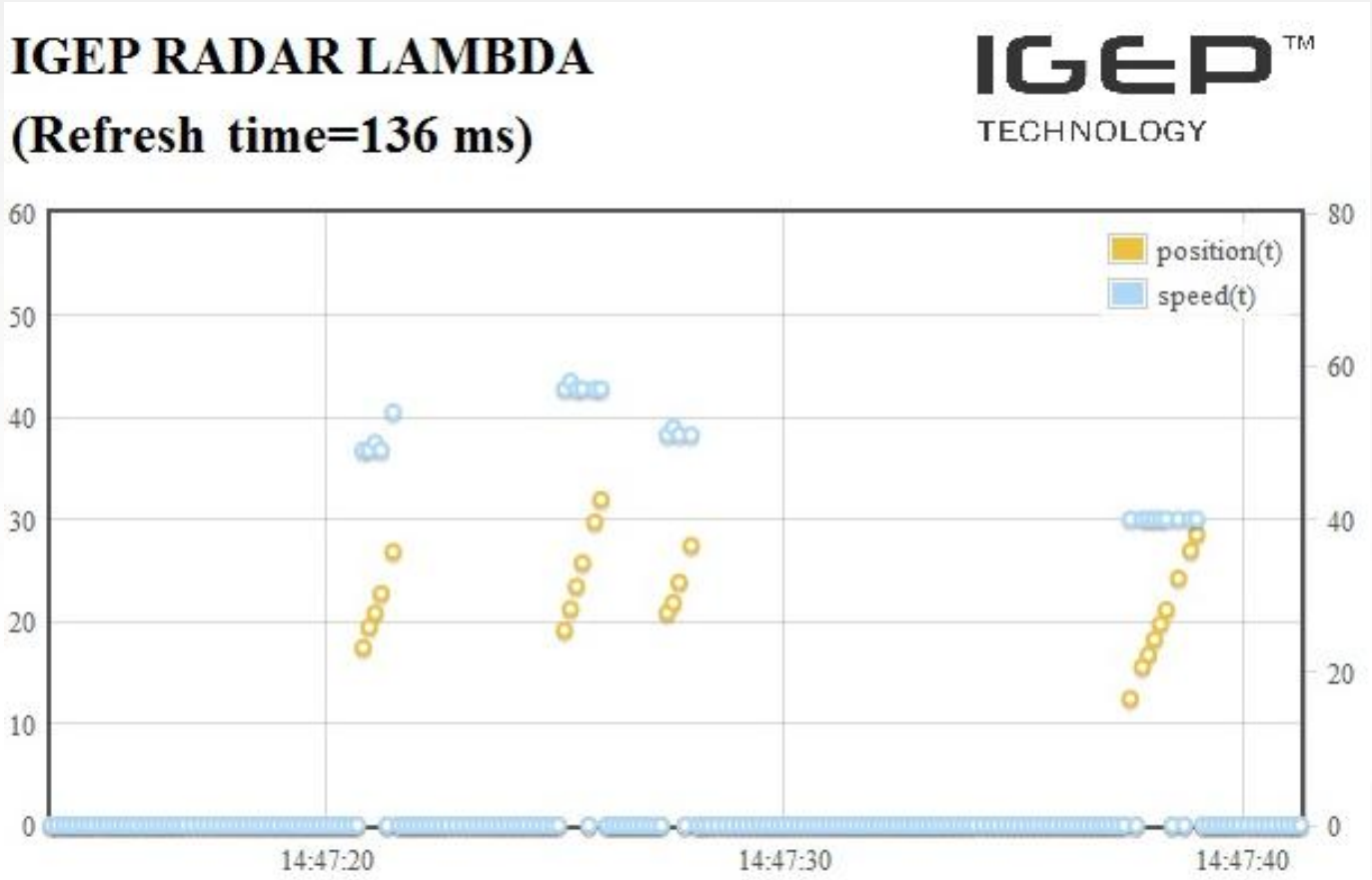
HIGHLIGHTS: HTML5 + AJAX Visualization

- **Gnuplot** canvas - http://gnuplot.sourceforge.net/demo_canvas/



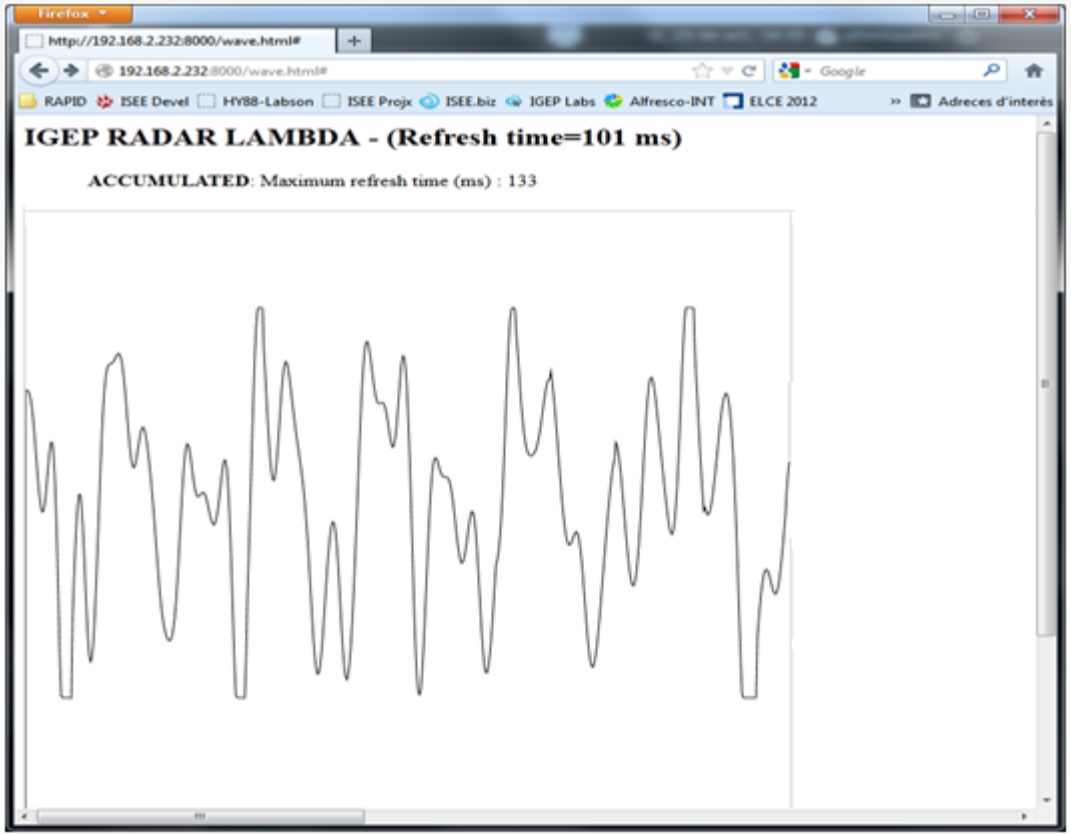
HIGHLIGHTS: HTML5 + AJAX Visualization

- **Flop** (jQuery extension) - http://www.flotcharts.org/



HIGHLIGHTS: HTML5 + AJAX Visualization

- raw **Canvas** HTML5 + AJAX - http://www.w3schools.com/ajax/
  - https://developer.mozilla.org/docs/Canvas_tutorial/



HIGHLIGHTS: Developing /time costs

- Directly C-code implementation from **developer**
  - Signal processing model is reimplemented by developer
    - Too much time (slow)
  - Poor accuracy in comparison to model
  - Knowledge about radar signal processing is needed
- **Automated generation** of C-code
  - Signal processing model tool generates C-code
    - The fastest way
  - Optimization depends on synthesizers and compiler tools
    - No specific hardware layout for DM3730
- Poor Linux infrastructure for TI C64x+ DSP core

DEMO

- IGEP v2 board (with DM3730 at 1 GHz)
- IGEP RADAR sensor
- HTML5 web browser running on Smartphone or Tablet

FUTURE

- DSP (DM3730 built-in TI C64x+ DSP core)
  - FFT Processing
  - ADC direct acquisition (change to McBSP interface)
- NEON coprocessor extension on ARM
  - libfftw3 support
  - memcpy optimizations
- Improve HTML5 web page
  - functionality
  - style / design

SUMMARY (1/2)

- Device based on Cortex A8 at 1GHz, Linux kernel 2.6.37 and poky
- Difficulties
  - Use built-in DSP core for poor software infrastructure in Linux
  - Too much developing time to take advantage of specific hardware and advanced software libraries

SUMMARY (2/2)

- Goals
  - Implemented software radar signal processing (FFT,...) without expensive hardware acceleration (like traditional FPGA integrated circuits). Automated C-code generation from radar model.
  - ARM architecture and a few electronic devices get an energy-efficient radar


ありがとう
謝謝
Danke schön
3Q
Thank you
Gràcies
Gracias
Dank U wel
Merci Beaucoup
Moito Brigado
감사합니다